Behavioral Cloning Project

The goals / steps of this project are the following:
* Use the simulator to collect data of good driving behavior
* Build, a convolution neural network in Keras that predicts steering angles from images
* Train and validate the model with a training and validation set
* Test that the model successfully drives around track one without leaving the road
* Summarize the results with a written report

Rubric Points
###Here I will consider the [rubric points](https://review.udacity.com/#!/rubrics/432/view) individually and describe how I addressed each point in my implementation.

###Files Submitted & Code Quality

####1. Submission includes all required files and can be used to run the simulator in autonomous mode
My project includes the following files:

* model.py: containing the script to create and train the model
* drive.py: For driving the car in autonomous mode
* model.h5: containing a trained convolution neural network
* writeup_report: .writeup_RyanKangt.pdf summarizing the results

####2. Submission includes functional code

Using the Udacity provided simulator and model file which is designed by me; the car can be driven autonomously around the track without crashing or deviating from the road. I generate "model.h5" file and ran it with "python drive.py model.h5".

####3. Submission code is usable and readable

The model.py file contains the code for training and saving the convolution neural network. The file shows the pipeline I used for training and validating the model, and it contains comments to explain how the code works.
When it comes to "model.py", I did not use generator in that model because the model without generator provides much better driving behavior even without memory issue. Even though the model with generator shows the lower training/validation MSE (about 0.00xx), the overall driving performance was poor (Dive to the river and climb to the mountain) In this files, I just included "model(generator).py" file which is the my model using generator for your inference. In addition, the processing speed of model with generator is much slower. I used 32 batch-size for generator. I cannot figure out yet why the performance of model with generator is worse even though I used the same CNN model, parameter and so on with the model without generator.

###Model Architecture and Training Strategy

####1. An appropriate model architecture has been employed

I started with very simple model, one Flatten and one Full connected layer model. Because of poor performance, I used conventional Lenet model. However, the driving behavior was not smooth and sometimes looks weird. Finally, I have referred to Nvidia model which is introduced in the class. My model consists of 3 convolution neural networks with 5x5 filters, 1 convolution neural network with 3x3 filters. The depths are 24, 36, 48 and 64 in order. In addition, I have used 3 fully connected layers after that. Each layer includes RELU layers to introduce nonlinearity, and the data is normalized in the model using a Keras Lambda layer. Finally, I have used cropping layer to remove unnecessary images.

####2. Attempts to reduce overfitting in the model

Every layer contains dropout layers with 0.5 in order to reduce overfitting. In addition, I reduced one convolutional layer and one full-connected layer from original Nvidia model to reduce overfitting. Finally, I gathered much data which are mainly “smooth driving for curved road”, “weaving back to the center of the road” for preventing overfitting. I separated training data and validating data as 8:2 ratios.

####3. Model parameter tuning

The model used an adam optimizer, so the learning rate was not tuned manually.

####4. Appropriate training data

I gathered the training data using PS4 analog joystick because it provides smoother vehicle movement and so, more elaborate steering data. I collected 2 laps of center driving images on the road and 2 laps of reversed direction (clock-wise) center driving images. In addition, I gathered 1 lap of smooth curve road driving image and 1 lap of weaving(recovering) back to the center road from out of the road. Therefore, I collected total about 11,000 data. I augmented them three times by using left and right camera images. In addition, I augmented them again two times by flipping the images. Overall, I could generalize the data (which is not biased to one side) with about 55,000 data and got natural autonomous driving pattern.

###Model Architecture and Training Strategy

####1. Solution Design Approach

In this project, I believe that the most important part is to collect well-balanced driving image/steering data. So, I gathered the data as explained above paragraph. Next step is to use a convolution neural network model. I have used a variety of model from very simple model to complex (e.g. Nvidia) model. I concluded that Nvidia model was the best one for my model even though I dropped a couple of layer and added aggressive dropout layer in order to reduce overfitting. Data augmentation was one of the important parts in this project. By increasing and generalizing the data at the same time through it, I can acquire great performance at the autonomous driving simulation. In the middle of designing my model, I checked how well the car was driving around the track. There was a couple of point where the car is deviating from the road. In that case, I first checked the MSE of validation/training data to figure out the status of overfitting/underfitting. In that case, I modified the model and parameters to reduce the overfitting/underfitting. In other cases, I collected more data in that point where the vehicle is deviating from the road. By teaching how the vehicle should turn itself back to road to the vehicle, I could finalize this project well.

####2. Final Model Architecture

The final model architecture (model.py lines 18-24) consisted of a convolution neural network with the following full connected layers. Please refer to the comment at the “model.py” file.

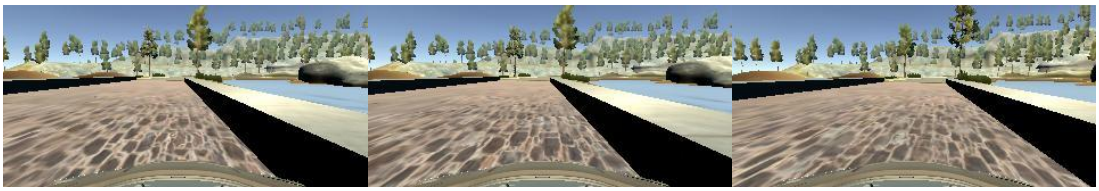
####3. Creation of the Training Set & Training Process (Revised)

At first, I gathered first 2 laps on track of center lane driving. Below is an example (Center Camera)
(I also gathered reverse-direction center driving data)

0.0

0.1188574

Then, I collected 1 more lap of smooth curving data in order to strengthen learning when car drive curve road.
Finally, I gathered several backing (recovering) data from the side of the road to center of the road as below.



After gathering all the data, I have augmented 3 times of data by using center, right, left camera.
By correcting fixed value (In here, +- 0.2) for left/right camera image, I could make more robust model.

0.0

0.2

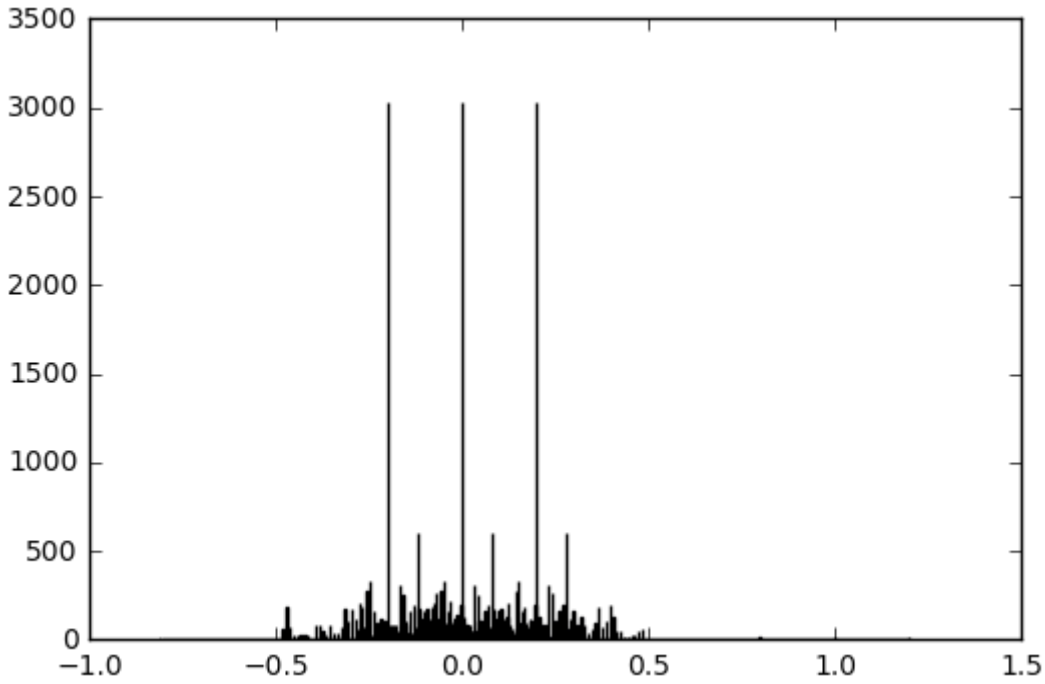
-0.2

Finally, I flipped the image for better generalization even though I already drive the vehicle at reverse-side for generalization. Example of image flipping is as below

-0.2

0.2

Below is the steering angle distribution. Because of data generalization & augmentation, the steering angle value and frequency are almost symmetry even though data of right turn angle is a little bit more. Therefore, we can expect that it would not be biased toward one direction. However, it is possible that the steering angle would be hold zero or too sharply change, even not in that case, because the steering angle is mostly concentrated on 0, -0.2, 0.2. However, relatively high frequency of 0.1, -0.1 in my model would protect this. To make better model, the distribution should be more close to Gaussian distribution for better driving behavior.



After gathering all the data, I finally randomly shuffled the data and put 20% of data into a validation set. By comparing the MSE of training/validation data set, I could figure out how much the model is overfitted or underfitted and so, modify the model to optimize it.